

APPLICATION  
FOR  
UNITED STATES LETTERS PATENT

TITLE: PACKET PROCESSING IN A ROUTER ARCHITECTURE  
APPLICANT: SATYENDRA YADAV, KAPIL JAIN AND ANAND  
RANGARAIAAN

CERTIFICATE OF MAILING BY EXPRESS MAIL

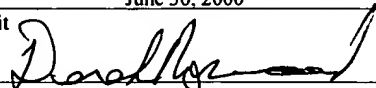
Express Mail Label No. EL558600466US

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

June 30, 2000

Date of Deposit

Signature



Derek W. Norwood

Typed or Printed Name of Person Signing Certificate

# PACKET PROCESSING IN A ROUTER ARCHITECTURE

## BACKGROUND

This invention relates to packet processing in a router architecture.

5        A conventional single-box router processes packets that need to be switched (or forwarded) as well as packets that contain control information called control packets for computing routing tables. A routing table keeps track of routes to particular network destinations. Packet switching  
10 includes the sending of data information in packets, through a network, to a remote location. Each packet of data information has a unique source address and carries its own destination address.

15        A conventional router has a device-driver that controls network interface cards located in the router. The device-driver handles communication between the interface cards between remotely located devices and a networking stack in the router. The networking stack is a stack of software layers such as IP (internet protocol), TCP/UDP (transmission control  
20 protocol/user datagram protocol), and socket library (application programming interface library) between networking applications and the device-driver. A packet traverses the

networking stack to be delivered to a networking application, which handles the processing of control packets of a networking device.

5

## DESCRIPTION OF DRAWINGS

FIGS. 1 and 2 illustrate receiving and transmitting a packet in a distributed router architecture according to an embodiment of the invention.

10

## DETAILED DESCRIPTION

15

A router architecture described below separates the processing of packet switching (or forwarding) from the processing of control packets. FIG. 1 shows an implementation of a distributed router architecture 1000 in which switches 100, 200 for handling packet switching and a controller 500 for processing control packets are distributed across a network such as an Ethernet network 50. The controller 500 controls the switches 100, 200.

20

The distributed router architecture 1000 may be made compatible with networking applications 502, such as telnet, that work in a conventional router. One way is by keeping the distributed switches 100, 200 as hidden from the networking applications 502. Otherwise, the networking applications 502

would require modification to work under the distributed architecture. Hiding the fact that the switches 100, 200 are distributed can provide a single router view of the switches 100, 200 to the networking applications 502. From the perspective of the networking applications 502, packets may appear to come from a router and not through a separate switch.

A known single-box router contains network interfaces, each controlled by a device-driver. In the distributed router architecture 1000, the network interfaces that communicate with other routers or hosts are placed in the switches 100, 200, but not in the controller 500. FIG. 1 shows network interfaces A, B, and C located in the switch 100 and network interfaces D and E located in the switch 200. To "hide" the fact that the switches are distributed over the Ethernet 50, a pseudo device-driver 512 can be introduced in the controller 500. The pseudo device-driver 512 causes control packets, which are packets containing routing information for computing routing tables, to be delivered to the networking stack (e.g. socket library 504, TCP/UDP 506, and IP 510) of the controller 500 as if coming from virtual interfaces Av, Bv, Cv, Dv, or Ev in the pseudo device-driver 512. Each virtual interface corresponds to an interface present in a switch. For example, the virtual interface Av in the controller 500 corresponds to

the interface A in the switch 100, and so on. To the networking applications 502, the controller 500 appears to be a conventional single-box router with as many interfaces as there are in all the switches.

5        A control packet that is received by one of switch's interfaces can be relayed to the controller 500 and appear as input to its corresponding virtual interface. Conversely, a control packet sent out through one of the virtual interfaces in the controller 500 can be relayed to an appropriate switch  
10        containing the corresponding interface. The receiving and sending of packets as described above can be accomplished using an IP encapsulation protocol termed a virtual interface protocol (VIP). The VIP is implemented by a VIP module 508 located in the controller 500 and VIP modules 104 and 204  
15        located in the switches 100 and 200, respectively. The encapsulation carries frames of one protocol which includes as the data in another protocol.

FIG. 1 illustrates how a control packet is received by the switch 100 that appears to reside in the controller 500  
20        and delivered to a networking application 502. In phase 1, the switch 100 receives a packet through the interface C. In phase 2, the switch 100 appends an external IP header to the packet and delivers the packet to the IP module 106. A protocol field in the external IP header is filled with

information regarding the VIP. The external header provides the destination address of the controller 500. In phase 3, the IP module 106, which has access to a routing table, performs route lookup for the controller 500 and sends the packet through an interface 102 specified in the routing table. In phase 4, the packet reaches the controller 500 at an interface 514 from the interface 102 and is delivered to the interface CV in the pseudo device driver 512. In phase 5, the interface Cv in the pseudo device-driver 512 delivers the packet to the IP module 510. In phase 6, the IP module 510 delivers the packet to a VIP module 508 based on the protocol field in the external IP header. In phase 7, the VIP module 508 strips off the external IP header and delivers the packet to the IP module 510. The packet is delivered to the IP module 510 as if it were coming from the virtual interface Cv corresponding to the interface C in the switch 100. In phases 8 to 10, the packet traverses the networking stack--IP 510, TCP/UDP 506, and Socket library 504--and the data is delivered to the application 502.

FIG. 2 illustrates how a control packet is transmitted from a networking application 502. In phases 11 to 13, a packet is written by the application 502 using the socket library 504 and gets encapsulated by the TCP/UDP module 506. It is then given to the IP module 510. In phase 14, the IP

module 510 appends an IP header and performs route lookup for the destination and sends it through the appropriate interface in the pseudo device-driver specified in the routing table. A conventional single-box router software can be used to compute the routing table for the pseudo or actual interfaces of distributed switches. In FIG. 2, the interface Ev is assumed to be the appropriate virtual interface. In phase 15, the pseudo device-driver 512 appends an external IP header to the packet. The destination address contained in the external IP header is that of the switch 200 that contains the interface E, which corresponds to the virtual interface Ev. The protocol field of the external IP header is filled with information regarding the VIP. The pseudo device-driver 512 performs a route lookup for the destination switch 200 and sends the packet to the switch 200 through the interface 514. In phase 16, the packet reaches the switch 200 at an interface device-driver 202. In phase 17, the interface device-driver 202 delivers the packet to an IP module 206. The IP module 206 in phase 18 sends the packet to a VIP module 204 based on the protocol field in the external IP header. In phase 19, the VIP module 204 strips off the external IP header. It then looks into the internal IP header for a source IP address. This source IP address corresponds to one of the network interfaces, the network interface E. In phase 20, the packet

is sent out through the corresponding interface E to a network interface.

In the foregoing example, the controller 500 and the switches 100, 200 were distributed across the Ethernet 50.

5 However, the controller 500 and the switches 100, 200 can be tightly coupled using a high-speed switching fabric instead of being distributed. The routing architecture and the method described above can also be employed for the tightly coupled controller and the switches.

10 Routing architectures represented by the distributed architecture 1000 and the tightly coupled architecture are scalable, provide a single point of management at the controller, and reduce the load on the network. Additionally, the logic circuit of the switches 100, 200 can be made simpler  
15 and more efficient. The foregoing distributed architecture can be implemented in any Network Operating System (NOS) without any changes to its networking stack and routing infrastructure such as routing protocols or a routing table manager. Any networking application that works in an NOS will  
20 work without any changes in the exemplary disclosed architecture. The distributed architecture would also be useful in creating a number of applications such as firewalls for distributed switches with the feature of single point management.



The foregoing techniques can be implemented, for example, in a computer program executable on a computer. The computer program can be stored on a storage medium, such as random access memory (RAM), readable by a general or special purpose programmable computer, for routing packets under the exemplary disclosed architecture.

5

Other implementations are within the scope of the following claims.